Xmgrace Basic Plotting Example

Xmgrace module, tutorial on creating a page with two graphs, adding plots to the graph, positioning graphs, adding titles and legends, and setting linestyles and colors, adding error bars. The data is generated using Numeric module.

Objective:Â We are going to create a portrait page with 2 plots, first plot on top will be composed of two curves.

You can ***download***Â the full source code. To run the source code at the command line, type: *"python xmgrace_tutorial_Numeric.py"*.

First lets create some data to plot

```
#import module MA
import MA

# First let's create some data
npoints=50
X=MA.arrayrange(npoints,typecode=MA.Float)
X=X/float(npoints)*2.*MA.pi

Y1=MA.sin(X)
Y2=MA.cos(X)
Y3=MA.tan(X
```

and set a current directory (it will be needed when we save the plots to the output files).

```
TEMPDIR = './'
```

Now let's import xmgrace module

```
# xmgrace module is inside the genutil package
from genutil import xmgrace
```

and initialize xmgrace

```
# create our xmgrace object
x=xmgrace.init()
```

The picture below shows the effect of this command – the initialization of the xmgrace window

First let's set our graphs area, for the two plots we need two graphs initialized. Also we'll make the page a portrait page.

```
# First let's set our graphs area
# graph 0, exist by default, but we need to add 1 graph, therefore:
x.add_graph() # adds one graph to our graph list (x.Graph)
# Let's change the orientation of the page to portrait
x.portrait()
```

Set the area for the graph 0 and the graph 1

```
# Now let's set the area of graph 0
x.Graph[0].vymin=.55  # starts at 55% of the page
x.Graph[0].vymax=.9   # stops at 90% of the page
x.Graph[0].vxmin=.1  # starts at 10% of the page
x.Graph[0].vxmax=.75   # stops at 75% of the page
# and the area of graph 1
x.Graph[1].vymin=.1 # starts at 10% of the page
x.Graph[1].vymax=.45 # 45 % of page
# Let's offset the 2 graphs just for fun
x.Graph[1].vxmin=.25  # starts at 55% of the page
x.Graph[1].vxmax=.9   # stops at 90% of the page
```

Set some titiles and subtitles for the graphs

```
# let's set the titles and subtitles
x.Graph[0].title='TRIGO'
x.Graph[0].stitle='SIN and COS'
x.Graph[1].stitle='TAN'
```

And the Y axis ranges for both graphs together with the tick marks

```
# ok now let's set the min ans max for the Y axis
x.Graph[0].yaxis.min=-1. # ymin for graph 0
x.Graph[0].yaxis.max=1. # ymax for graph 0
x.Graph[1].yaxis.min=-1.5 # ymin for graph 1
x.Graph[1].yaxis.max=1.5  # ymax for graph 1

# Now let's set the tick marks for Graph 0
x.Graph[0].yaxis.tick.inc=1 # Main tick every unit
x.Graph[0].yaxis.tick.minor_ticks=4 # 4 sub in between , 1 every .25 units
# Now let's set the tick marks for Graph 1
x.Graph[1].yaxis.tick.inc=10 # Main tick every 10
x.Graph[1].yaxis.tick.minor_ticks=4 # 4 sub in between , 1 every 2.5 units
```

Set X axis and it's tick marks

```
# X values are between 0 and 2pi
# therefore let's set the axis  from -1,1
x.Graph[0].xaxis.min=0.
x.Graph[0].xaxis.max=2.*MA.pi
x.Graph[1].xaxis.min=0.
x.Graph[1].xaxis.max=2.*MA.pi

# Set the xaxis ticks
dic={0.:'0',MA.pi/2:'PI/2',MA.pi:'PI',3*MA.pi/2:'3PI/2',2*MA.pi:'2PI'}
x.Graph[0].xaxis.tick.spec.loc=dic
# way 2 the "less convential" way
x.Graph[1].xaxis.tick.spec.loc=dic
# Finally let's reduce the size a bit
x.Graph[0].xaxis.tick.label.char_size=.8
x.Graph[1].xaxis.tick.label.char_size=.8
```

Set the legends for both graphs

```
# now the legend
x.Graph[0].legend.char_size=.8
x.Graph[0].legend.x=.8  # Legend at 80% in x
x.Graph[0].legend.x=.8  # Legend at 80% in y
#or
x.Graph[1].legend.char_size=.8
x.Graph[1].legend.x=.05
```

```
x.Graph[1].legend.y=.35
```

We want to have two curves (sets) on one graph, by default we have 1 setÂ per graph, so we need to add another one toÂ theÂ graph 0

```
# Ok now let's play with the appearence of the sets themself
# Ok by default we only have 2 sets (1 per graph)
# So we need to add 2 sets
x.add_set(0) # adding to graph 0 (since it's graph 0, the 0 is optional)
```

Now lets set the line widths, colors and linestyles

```
# First let's change the line width
x.Graph[0].Set[0].line.linewidth=2
x.Graph[0].Set[1].line.linewidth=2
x.Graph[1].Set[0].line.linewidth=2
# Let's set the colors
x.Graph[0].Set[0].line.color='red' # xmgrace default colors are coded
x.Graph[0].Set[1].line.color=4 # color 4 is blue
# Let's define a new color for the tangeante
x.add_color('purple')
x.Graph[1].Set[0].line.color='purple' # use the color we just defined
# Let's set the second's set style to dash
x.Graph[1].Set[0].line.linestyle='solid'
x.Graph[0].Set[1].line.linestyle='dash'
x.Graph[0].Set[0].line.linestyle=2 # 2 is dot
```

and define the legends for each set

```
# Now let's set the name for each set, to be used in the legend
x.Graph[0].Set[0].legend='SIN'
x.Graph[0].Set[1].legend='COS'
x.Graph[1].Set[0].legend='TAN'
```

We would like to have a zero line plotted, so let's add that

```
# ok one last trick, in order to have the 0 line plotted
# let's create a dummy array with zeros in it:
zero=[0,0]
# add a dataset and assign it graph 1 black color
x.add_set(1,'black')
```

Also let's play with adding some text to the page

```
# Finally just for fun let's place a big red "Sample" accross it
x.add_string(0.5,0.5,'Sample',color='red',char_size=9,rot=55,just=14)
```

Now we can plot all the graphs

```
# And plot these babies:
x.plot([Y1,Y2,Y3,zero],xs=[X,X,X,[0.,2.*MA.pi] ])
# You MUST pass a list of slab, even if only one slab
# or you can plot then 1 by 1
```

Here is the resulting page

We will improve on the plots by adding an error bars.
First we need to clean up the plots

```
# first let's clear
x('kill G0.S0')
x('kill G0.S1')
x('kill G1.S0')
x('kill G1.S1')
```

Let' s add the mask to the tanget

```
# Now the tan is pretty ugly because of extreme let's mask everything
# that is greater than 1.5 and redraw that
Y3=MA.masked_greater(Y3,1.5)
```

and error bars for all the plots

```
# Also we're going to add error bars 10% of the value
# dx for the sin
YY1=MA.zeros((2,npoints),typecode=MA.Float)
YY1[0]=Y1
YY1[1]=Y1*.1
x.Graph[0].Set[0].type='xydx'
x.Graph[0].Set[0].errorbar.status='on'
x.Graph[0].Set[0].errorbar.color='red'
# dy for the cos
YY2=MA.zeros((2,npoints),typecode=MA.Float)
YY2[0]=Y2
YY2[1]=Y2*.1
x.Graph[0].Set[1].type='xydy'
x.Graph[0].Set[1].errorbar.status='on'
x.Graph[0].Set[1].errorbar.color=x.Graph[0].Set[1].line.color
# dy and dx for the tan
YY3=MA.zeros((3,npoints),typecode=MA.Float)
YY3[0]=Y3
YY3[1]=Y3*.1
YY3[2]=Y3*.1
x.Graph[1].Set[0].type='xydxdy'
x.Graph[1].Set[0].errorbar.status='on'
x.Graph[1].Set[0].errorbar.color='purple'
```

Now we can plot it again. We will plot each set separately, first we will plot first set (set 0) on graph 0

```
#plot first set from first graph
x.plot(YY1,xs=X,G=0,S=0)
```

Here is the result. Notice that we didÂ not clear the text.

and add the second set (set 1) to graph 0

```
#plot second set from first graph
x.plot(YY2,xs=X,G=0,S=1)
```

The result looks like that

Finally let's put the second graph and the first set (tangent line)

```
#plot first set from second graph
x.plot(YY3,xs=X,G=1,S=0)
```

and add the zero line as a second set to the second graph

```
# Now since we are passing lists and not MAs we need to wrap them into a list
x.plot([zero],xs=[[0.,2.*MA.pi] ],G=1,S=1)
```

Here is our final page

Now we can save it to the different output file formats as follows:

```
# Finally let's save the result
x.ps(TEMPDIR+'xmgrace_demo_Numeric') # postscript
x.ps(TEMPDIR+'xmgrace_demo_Numeric_gray',color='grayscale') # grayscale postscript
x.jpeg(TEMPDIR+'xmgrace_demo_Numeric',dpi=300,quality=80) # jpeg 300dpi, 80% quality compression
x.pdf(TEMPDIR+'xmgrace_demo_Numeric')
x.eps(TEMPDIR+'xmgrace_demo_Numeric')
x.svg(TEMPDIR+'xmgrace_demo_Numeric')
x.mif(TEMPDIR+'xmgrace_demo_Numeric')
x.pnm(TEMPDIR+'xmgrace_demo_Numeric')
x.png(TEMPDIR+'xmgrace_demo_Numeric')
x.metafile(TEMPDIR+'xmgrace_demo_Numeric')
```

Â Lastly we will clean up by closing the xmgrace window.

```
# Finish
x.close()
```